
Subject: [PATCH v4 06/19] slab/slub: struct memcg_params
Posted by [Glauber Costa](#) on Fri, 12 Oct 2012 13:41:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

For the kmem slab controller, we need to record some extra information in the kmem_cache structure.

Signed-off-by: Glauber Costa <glommer@parallels.com>
Signed-off-by: Suleiman Souhlal <suleiman@google.com>
CC: Christoph Lameter <cl@linux.com>
CC: Pekka Enberg <penberg@cs.helsinki.fi>
CC: Michal Hocko <mhocko@suse.cz>
CC: Kamezawa Hiroyuki <kamezawa.hiroyu@jp.fujitsu.com>
CC: Johannes Weiner <hannes@cmpxchg.org>
CC: Tejun Heo <tj@kernel.org>

include/linux/slab.h | 25 +++++
include/linux/slab_def.h | 3 +++
include/linux/slub_def.h | 3 +++
mm/slab.h | 13 +++++
4 files changed, 44 insertions(+)

```
diff --git a/include/linux/slab.h b/include/linux/slab.h
index 0dd2dfa..e4ea48a 100644
--- a/include/linux/slab.h
+++ b/include/linux/slab.h
@@ -177,6 +177,31 @@ unsigned int kmem_cache_size(struct kmem_cache *);
#define ARCH_SLAB_MINALIGN __alignof__(unsigned long long)
#endif
```

```
+ #include <linux/workqueue.h>
+ /*
+  * This is the main placeholder for memcg-related information in kmem caches.
+  * struct kmem_cache will hold a pointer to it, so the memory cost while
+  * disabled is 1 pointer. The runtime cost while enabled, gets bigger than it
+  * would otherwise be if that would be bundled in kmem_cache: we'll need an
+  * extra pointer chase. But the trade off clearly lays in favor of not
+  * penalizing non-users.
+  *
+  * Both the root cache and the child caches will have it. For the root cache,
+  * this will hold a dynamically allocated array large enough to hold
+  * information about the currently limited memcgs in the system.
+  *
+  * Child caches will hold extra metadata needed for its operation. Fields are:
+  *
+  * @memcg: pointer to the memcg this cache belongs to
+  */
+ struct memcg_cache_params {
```

```

+ bool is_root_cache;
+ union {
+ struct kmem_cache *memcg_caches[0];
+ struct mem_cgroup *memcg;
+ };
+};
+
+/*
+ * Common kmalloc functions provided by all allocators
+ */
diff --git a/include/linux/slab_def.h b/include/linux/slab_def.h
index 36d7031..665afa4 100644
--- a/include/linux/slab_def.h
+++ b/include/linux/slab_def.h
@@ -81,6 +81,9 @@ struct kmem_cache {
+/*
+ int obj_offset;
+ #endif /* CONFIG_DEBUG_SLAB */
+ #ifdef CONFIG_MEMCG_KMEM
+ struct memcg_cache_params *memcg_params;
+ #endif

/* 6) per-cpu/per-node data, touched during every alloc/free */
/*
diff --git a/include/linux/slub_def.h b/include/linux/slub_def.h
index df448ad..961e72e 100644
--- a/include/linux/slub_def.h
+++ b/include/linux/slub_def.h
@@ -101,6 +101,9 @@ struct kmem_cache {
+ #ifdef CONFIG_SYSFS
+ struct kobject kobj; /* For sysfs */
+ #endif
+ #ifdef CONFIG_MEMCG_KMEM
+ struct memcg_cache_params *memcg_params;
+ #endif

+ #ifdef CONFIG_NUMA
+ /*
diff --git a/mm/slab.h b/mm/slab.h
index 66a62d3..5ee1851 100644
--- a/mm/slab.h
+++ b/mm/slab.h
@@ -92,4 +92,17 @@ void get_slabinfo(struct kmem_cache *s, struct slabinfo *sinfo);
void slabinfo_show_stats(struct seq_file *m, struct kmem_cache *s);
ssize_t slabinfo_write(struct file *file, const char __user *buffer,
size_t count, loff_t *ppos);
+
+
+ #ifdef CONFIG_MEMCG_KMEM

```

```
+static inline bool is_root_cache(struct kmem_cache *s)
+{
+ return !s->memcg_params || s->memcg_params->is_root_cache;
+}
+#else
+static inline bool is_root_cache(struct kmem_cache *s)
+{
+ return true;
+}
+
+#endif
#endif
--
1.7.11.4
```
